UNITED STATES PATENT AND TRADEMARK OFFICE

17-A

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/798,913	03/11/2004	Jeffrey Michael Davis	ROC920030411US1	7004

46797 7590 02/12/2007
IBM CORPORATION, INTELLECTUAL PROPERTY LAW
DEPT 917, BLDG. 006-1
3605 HIGHWAY 52 NORTH
ROCHESTER, MN 55901-7829

EXAMINER	
WANG, JUE S	
ART UNIT	PAPER NUMBER
2109	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	02/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)
	10/798,913	DAVIS ET AL.
	Examiner	Art Unit
	Jue S. Wang	2109

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11 March 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *11 March 2004* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed March 11, 2004.

2. Claims 1-20 have been examined.

Specification

3. The specification is objected to as failing to provide proper antecedent basis for the claimed subject matter. See 37 CFR 1.75(d)(1) and MPEP § 608.01(o). Correction of the following is required:

The "template agent" of claim 9 lacks antecedent basis in the specification. The specification discloses template agents providing functionalities directed to managing background processes (see page 3, paragraph [0009], and page 11, paragraph [0036]). The specification does not disclose the limitations recited in claim 9 which states that template agents comprise a first template providing functionality directed to presenting and managing data, a second template providing functionality directed to presenting database records, and a third template providing functionality directed to coordinating transactions among different components of an application.

4. The specification is objected to because of the following minor informalities:

Page 3, paragraph [0009], line 1, the word "environment" should be "embodiment". Appropriate correction is needed.

Page 11, paragraph [0036], the figure label 211A for template form, 211B for template view, 211C for template shared action, and 211D for template agent do not exist in Fig 2. The corresponding labels in Fig 2. are 211 for template form, 212 for template view, 213 for template shared action, and 214 for template agent. Appropriate correction is required.

Page 18, paragraph [0055], line 4, the phrase "solving particular types business problems" should be "solving particular types of business problems". Appropriate correction is required.

Page 20, paragraph [0062], line 2, the phrase "accordingly the following creation of an agent" should be "accordingly following the creation of an agent". Appropriate correction is required.

The use of the trademark JAVA and C++ has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

Drawings

5. The drawings are objected to because in Fig 6B, the label "TO 610" and "TO 602" should be "TO 605" instead. The specification discloses that the step following steps 613, 617, 643, and 646 should be step 605 (see page 19, paragraph [0061]). Additionally, step 646 should read "TIE FUNCTIONALITY TO AGENT" instead of "THE FUNCTIONALITY TO AGENT".

Corrected drawing sheets in compliance with 37 CFR 1.121(d) are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The figure or figure number of an amended drawing should not be labeled as "amended." If a drawing figure is to be canceled, the appropriate figure must be removed from the replacement sheet, and where necessary, the remaining figures must be renumbered and appropriate changes made to the brief description of the several views of the drawings for consistency. Additional replacement sheets may be necessary to show the renumbering of the remaining figures. Each drawing sheet submitted after the filing date of an application must be labeled in the top margin as either "Replacement Sheet" or "New Sheet" pursuant to 37 CFR 1.121(d). If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claim 9 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

As per claim 9, it recites the limitation that template agents comprise a first template providing functionality directed to presenting and managing data, a second template providing functionality directed to presenting database records, and a third template providing functionality directed to coordinating transactions among different components of an application. There is a lack of antecedent basis for this limitation in the specification where it is disclosed that template agents provide functionalities directed to managing background processes (see page 3, paragraph [0009], and page 11, paragraph [0036] of specification). Parent claim 8 also recited the limitation that template agents provide functionality directed to manage background processes. The functionalities recited in claim 9 are associated with front-end templates as disclosed in the specification (see page 3, paragraph [0009], and page 11, paragraph [0036] of specification) and recited in claims 4-7, 12-14, and 18-20. For compact prosecution of the claims, the office has interpreted the limitation as an error of what should have been "front-end template" instead of "template agent".

Claim Rejections - 35 USC § 101

8. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

9. Claims 1-20 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

Claim 1 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. In claim 1, an "application development environment" is recited; however, it appears that the application development environment would reasonably be interpreted by one of ordinary skill in the art as software, per se, since the development template suite and the framework components would reasonably be interpreted by one of ordinary skill in the art as software, per se. As such, it is believed that the application development of claim 1 is reasonably interpreted as functional descriptive material, per se.

Claims 2-7 fail to resolve the deficiencies of claim 1. Claims 2-7 merely disclose additional features of the application development environment of claim 1 and do not provide either a physical transformation or a useful, concrete and tangible result.

Claim 8 is rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. The current focus of the Patent Office in regard to statutory inventions under 35 U.S.C. § 101 for method claims and claims that recite a judicial exception (software) is that the claimed invention recite a practical application. Practical application can be provided by a physical transformation or a useful, concrete and tangible result. No physical transformation is recited: the steps of providing templates, receiving user selections, and implementing functionalities are merely instructions within a computer program. The claim merely recite the steps associated with using a standard application development template suite and does not recite a result, so it does not satisfy the requirement of producing a useful, concrete, and tangible result.

Claims 9 and 12-14 fail to resolve the deficiencies of claim 8. The limitations recited in claims 9 and 12-14 disclose additional features of the templates in the template suite recited in claim 8 and do not provide either a physical transformation or a useful, concrete and tangible result

Claims 10 and 11 fail to resolve the deficiencies of claim 8. The additional steps disclosed in claims 10 and 11 are also instructions within a computer program and do not provide either a physical transformation or a useful, concrete and tangible result.

With respect to claim 15, the "computer-readable medium," in accordance with applicant's specification, may be a signal-bearing medium (see page 6, paragraph [0022]). This subject matter is not limited to that which falls within a statutory category of invention because it is not limited to a process, machine, manufacture, or a composition of matter. Instead, it includes a form of energy. Energy does not fall within a statutory category since it is clearly not a series of steps or acts to constitute a process, not a mechanical device or combination of mechanical devices to constitute a machine, not a tangible physical article or object which is some form of matter to be a product and constitute a manufacture, and not a composition of two or more substances to constitute a composition of matter.

Claims 16-20 fail to resolve the deficiencies of claim 15; they merely disclose additional features of the application development program residing in the computer-readable medium of claims 15.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

11. Claims 1, 4, 6, and 7 are rejected under 35 U.S.C. 102(b) as being anticipated by Boudreau et al., "NetBeans: The Definitive Guide", 2002, (hereinafter NetBeans).

As per claim 1, NetBeans teaches an application development environment configured to facilitate development of an application (i.e., NetBeans is an open source integrated development environment, see page xiii), comprising:

a standard development template suite (i.e., NetBeans present prototype Java classes and other source entities as templates which can be instanced in the source tree as new files, see pages 39-41) comprising at least one template agent providing functionality directed to managing background processes of the application (i.e., file system templates providing backend functionality directed towards file management, see page 274-276 and 330-333) and at least one front-end template providing interface functionality to communicate with the background processes (i.e., front-end templates such as GUI and JSP templates, see page 39-41, 198, 199, 202, and 203);

a plurality of framework components comprising a front-end core script library containing code elements used to add front-end functionality to the application not provided by the front-end template and a back-end core script library containing code elements used to add functionality to the application not provided by the template agent (i.e., NetBeans provides public interfaces and classes to module writers that are divided into specific APIs for dealing with different types of functionality such as front-end presentation functionalities and back-end data processing functionalities, see pages 244-246); and a global script library containing data structures available for use by the framework components in creating the front-end and back-end application components (i.e., Sun Microsystems' JDK consisting of fundamental Java class libraries are available to NetBeans users, see page 2).

As per claim 4, NetBeans teaches the application development environment of claim 1, which has been addressed. NetBeans further teaches that at least one of the front-end templates of the application development environment provides functionality directed to presenting and managing data (i.e., the template for TopComponent, the fundamental GUI component of NetBeans, see pages 298, 299, and 313).

As per claim 6, NetBeans teaches the application development environment of claim 1, which has been addressed. NetBeans further teaches that at least one of the front-end templates of the application development environment provides functionality directed to coordinating transactions among different component of an application (i.e., the template for CallableSystemAction which performs basic actions for user invokable behavior, see pages 264-266, 285, 325, and 326).

As per claim 7, NetBeans teaches the application development environment of claim 1, which has been addressed. NetBeans further teaches that at least one of the front-end templates of the application development environment comprises at least two of: a template providing functionality directed to presenting and managing data; a template providing functionality directed to presenting database records; and a template providing functionality directed to coordinating transactions among different components of an application (i.e., the template for TopComponent, the fundamental GUI component of NetBeans and the template for CallableSystemAction which performs basic actions for user invokable behavior, see pages 264-266, 285, 298, 299, 313, 325 and 326).

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 2 and 3 are rejected under 35 U.S.C. 103(a) as being unpatentable over Boudreau et al., "NetBeans: The Definitive Guide", 2002, (hereinafter NetBeans).

As per claim 2, NetBeans teaches the application development environment of claim 1, which has been addressed. NetBeans further teaches that the framework components comprise a project specific script library listing components from the front-end core script library for use with the application (i.e., the NetBeans Explorer presents views of the project including mounted jar files used with the project, where it is well known in the art that Java Archive (JAR) files are used to distribute library of classes providing some specific set of functionalities, see pages 29 and 30).

NetBeans does not specifically disclose that the project specific script library (i.e., JAR file associated with the project) is a front-end project specific script library. However, it is well known in the art that good software design following the principle of modularity dictates that software systems should be divided into a front-end and a back-end when applicable. Therefore, it is obvious that either the jar files themselves can be classified into providing specific front-end or back-end functionalities or the specific classes within the jar files are grouped based on the functionalities they provide.

As per claim 3, NetBeans teaches the application development environment of claim 1, which has been addressed. NetBeans further teaches that the framework components comprise a project specific script library listing components from the back-end core script library use to create the application (i.e., the NetBeans Explorer presents views of the project including mounted jar files used with the project, where it is well known in the art that Java Archive (JAR) files are used to distribute library of classes providing some specific set of functionalities, see pages 29 and 30).

NetBeans does not specifically disclose that the project specific script library (i.e., JAR file associated with the project) is a back-end project specific script library. However, it is well known in the art that good software design following the principle of modularity dictates that software systems should be divided into a front-end and a back-end when applicable. Therefore, it is obvious that either the jar files themselves can be classified into providing specific front-end or back-end functionalities or the specific classes within the jar files are grouped based on the functionalities they provide.

14. Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Boudreau et al., "NetBeans: The Definitive Guide", 2002, (hereinafter NetBeans), in view of Saimi et al., (US 2001/0047402 A1, published: 11/29/2001).

As per claim 5, NetBeans teaches the application development environment of claim 1, which has been addressed.

NetBeans further teaches that at least one of the front-end templates of the application development environment is a template for JSP, but it does not specifically teach that templates for JSPs provide functionality directed to presenting database records.

Saimi et al. is cited to teach a method for developing web applications by automatically generating code from templates selected from a template list, where JSP templates are used to generate JSP components that generates HTML pages to presents data retrieved by Beans from a database, see abstract, [0015], [0074], [0076], and [0085].

NetBeans and Saimi et al. are analogous art because they are both from the same field of endeavor of methods for application development. At the time of the invention, it would have been obvious to modify NetBeans with Saimi et al. since NetBeans provides users a means of creating customized templates (see page 41 of NetBeans) and it is desirable to provide interfaces to databases since databases are an essential element to most web based applications.

15. Claims 8-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saimi et al., (US 2001/0047402 A1, published: 11/29/2001), in view of et al. Sun Microsystems, "Building Web Components, Forte™ for Java™ Programming Series", 8/2001, (hereinafter Sun).

As per claim 8, Saimi et al. teaches a method comprising:

providing a standard development template suite (i.e., a method for developing web applications by automatically generating code from templates selected from a template list, see abstract and [0015]) comprising at least one template agent providing functionality directed to manage background processes (i.e., the Bean templates used to generate Beans that accesses databases on the server side, and performs business and data related processing, see [0074], [0076], and [0088]) and at least one front-end template providing interface functionality to configure the background processes action (i.e., the JSP templates used to generate JSP components that generates HTML pages to present and manage data retrieved from the database, and the servlet templates used to generate servlet components that executes requests from the client by issuing requests to the corresponding Bean and passing the Bean to the JSP so that data gathered from the Bean can be retrieved and presented, see [0074], [0076], [0080], and [0085]);

receiving user selections of at least one front-end template and at least one template agent (i.e., selection screens are presented for JSP, servlet, and Bean templates to facilitate the selection of the corresponding templates, see Fig 28, Fig 29, [0132], and [0133]);

implementing front-end functionality from the selected front-end template into an application (i.e., JSP and servlet source code generated based on the selected templates are incorporated into a web application, see Fig 6, Fig 9, [0080], [0085], and [0147]-[0149]); and

implementing back-end functionality from the selected template agent into the application (i.e., Bean source code generated based on the selected template are incorporated into a web application, see Fig 11, [0088], and [0147]-[0149]).

Saimi et al. does not teach that the method comprises implementing additional front-end functionality, not provided by the selected front-end template, to the application from a front-end core script library; and implementing additional back-end functionality, not provided by the selected template agent, to the application from a back-end core script library.

Sun is cited to teach a method of developing web components, including implementing additional front-end functionalities not provided by the selected front-end template, to the application from a front-end core script library; and implementing additional back-end functionalities, not provided by the selected template agent, to the application from a back-end core script library (i.e., adding custom actions to JSP pages from Java built-in tag libraries, where the tag libraries ietags, dbtags, and tptags provide both front-end and back-end functionalities not already implemented by standard actions available to JSP pages, see pages 13, 19, and page 43, Fig 2-11).

Saimi et al. and Sun are analogous art because they are both from the same field of endeavor of methods for web application development. At the time of the invention, it would have been obvious to modify Saimi et al. to include the feature of implementing from libraries, additional functionalities not provided by the templates as taught by Sun because Saimi et al. provide the means for the addition, extension or modification of the generated web application (see [0143] and [0148]).

As per claim 9, Saimi et al. in view of Sun teaches the method of claim 8, which has been addressed. Saimi et al. further teaches that the at least one front-end template comprises a first template providing functionality directed to presenting and managing data (i.e., the JSP templates used to generate JSP components that generates HTML pages to present and manage data retrieved from the database, see [0074], [0076], and [0085]), a second template providing functionality directed to presenting database records (i.e., the JSP templates corresponding to JSP components that generates HTML pages to present data retrieved by Beans from a database, see [0074], [0076], and [0085]), and a third template providing functionality directed to coordinating transactions among different components of an application (i.e., the servlet templates used to generate servlet components that executes requests from the client by issuing requests to the corresponding Bean and passing the Bean to the JSP so that data gathered from the Bean can be retrieved and presented, see [0074], [0076], and [0080]); and that the method comprises implementing functionality from the first, second, and third template into the application (i.e., JSP and servlet source code generated based on the selected templates are incorporated into a web application, see Fig 6, Fig 9, [0080], [0085], and [0147]-[0149]).

As per claim 10, Saimi et al. in view of Sun teaches the method of claim 8, which has been addressed.

Saimi et al. does not teach that implementing additional front-end functionality, not provided by the selected front-end template, to the application from a front-end core script library comprises implementing additional front-end functionality identified in a front-end project specific script library.

Sun is cited to teach a method of developing web components, including implementing additional front-end functionality, not provided by the selected front-end template, to the application from a front-end core script library by implementing additional front-end functionality identified in a front-end project specific script library (i.e., adding custom actions to JSP pages from custom tag libraries developed for specific projects to implement additional custom actions not provided by the built-in tag libraries, see pages 27-46). While Sun does not specifically disclose that the custom tag libraries are categorized as front-end and back-end, it is well known in the art that good software design following the principle of modularity dictates that software systems should be divided into a front-end and a back-end when applicable. Therefore, it is obvious that the custom tag libraries can be classified as providing front-end or back-end functionalities.

As per claim 11, Saimi et al. in view of Sun teaches the method of claim 8, which has been addressed.

Saimi et al. does not teach that implementing additional back-end functionality, not provided by the selected template agent, to the application from a back-end core script library comprises implementing additional backend functionality identified in a back-end project specific script library.

Sun is cited to teach a method of developing web components, including implementing additional back-end functionality, not provided by the selected back-end template, to the application from a back-end core script library by implementing additional back-end functionality identified in a back-end project specific script library (i.e., adding custom actions to JSP pages from custom tag libraries developed for specific projects to implement additional custom actions not provided by the built-in tag libraries, see pages 27-46). While Sun does not specifically disclose that the custom tag libraries are categorized as front-end and back-end, it is well known in the art that good software design following the principle of modularity dictates that software systems should be divided into a front-end and a back-end when applicable. Therefore, it is obvious that the custom tag libraries can be classified as providing front-end or back-end functionalities.

As per claim 12, Saimi et al. in view of Sun teaches the method of claim 8, which has been addressed. Saimi et al. further teaches that the template suite comprises at least one front-end template providing functionality directed to presenting and managing data (i.e., the JSP templates used to generate JSP components that generates HTML pages to present and manage data retrieved from the database, see [0074], [0076], and [0085]).

As per claim 13, Saimi et al. in view of Sun teaches the method of claim 8, which has been addressed. Saimi et al. further teaches that the template suite comprises at least one front-end template providing functionality directed to presenting database records (i.e., the JSP templates used to generate JSP components that generates HTML pages to present data retrieved by Beans from a database, see [0074], [0076], and [0085]).

As per claim 14, Saimi et al. in view of Sun teaches the method of claim 8, which has been addressed. Saimi et al. further teaches that the template suite comprises at least one front-end template providing functionality directed to coordinating transactions among different components of an application (i.e., the servlet templates used to generate servlet components that executes requests from the client by issuing requests to the corresponding Bean and passing the

Bean to the JSP so that data gathered from the Bean can be retrieved and presented, see [0074], [0076], and [0080]).

As per claim 15, Saimi et al. teaches a method for providing a standard development template suite comprising at least one template agent providing functionality directed to manage background processes and at least one front-end template providing interface functionality to configure the background processes (i.e., a method for developing web applications by selecting JSP, servlet, and Bean templates used to generate corresponding JSP, servlet, and Bean components, see abstract, [0015], [0074], [0076], [0080], and [0085]);

receiving user selections of one or more templates from the template suite (i.e., selection screens are presented for JSP, servlet, and Bean templates to facilitate the selection of the corresponding templates, see Fig 28, Fig 29, [0132], and [0133]); and

for each selected template:

determining if the selected template is a front-end template and, if so implementing front-end functionality from the selected front-end template into an application (i.e., for selected JSP and servlet templates, generate JSP and servlet source code to be incorporated into a web application, see Fig 6, Fig 9, [0080], [0085], and [0147]-[0149]); and

determining if the selected template is a template agent and, if so, implementing back-end functionality from the selected template agent into the application (i.e., for selected Bean templates, generate Bean source code to be incorporated into a web application, see Fig 11, [0088], and [0147]-[0149]).

Saimi et al. does not specifically disclose that the method is implemented as a program stored in a computer-readable medium, to be executed by a computer. However, it is obvious that one of ordinary skill in the art can implement the method taught by Saimi et al. as a program that can be stored in a computer-readable medium and executed by a computer.

Saimi et al. does not teach implementing additional front-end functionality, not provided by the selected front-end template, to the application from a front-end core script library, and implementing additional back-end functionality, not provided by the selected template agent, to the application from a back-end core script library.

Sun is cited to teach a method of developing web components, including implementing additional front-end functionalities not provided by the selected front-end template, to the application from a front-end core script library; and implementing additional back-end functionalities, not provided by the selected template agent, to the application from a back-end core script library (i.e., adding custom actions to JSP pages from Java built-in tag libraries, where the tag libraries ietags, dbtags, and tptags provide both front-end and back-end functionalities not already implemented by standard actions available to JSP pages, see pages 13, 19, and page 43, Fig 2-11).

As per claim 16, Saimi et al. in view of Sun teaches the computer-readable medium of claim 15, which has been addressed.

Saimi et al. does not teach that implementing additional front-end functionality, not provided by the selected front-end template, to the application from a front-end core script library comprises implementing additional front-end functionality identified in a front-end project specific script library.

Sun is cited to teach a method of developing web components, including implementing additional front-end functionality, not provided by the selected front-end template, to the application from a front-end core script library by implementing additional front-end functionality identified in a front-end project specific script library (i.e., adding custom actions to JSP pages from custom tag libraries developed for specific projects to implement additional custom actions not provided by the built-in tag libraries, see pages 27-46). While Sun does not specifically disclose that the custom tag libraries are categorized as front-end and back-end, it is well known in the art that good software design following the principle of modularity dictates that software systems should be divided into a front-end and a back-end when applicable. Therefore, it is obvious that the custom tag libraries can be classified as providing front-end or back-end functionalities.

As per claim 17, Saimi et al. in view of Sun teaches the computer-readable medium of claim 15, which has been addressed.

Saimi et al. does not teach that implementing additional back-end functionality, not provided by the selected template agent, to the application from a back-end core script library comprises implementing additional backend functionality identified in a back-end project specific script library.

Sun is cited to teach a method of developing web components, including implementing additional back-end functionality, not provided by the selected back-end template, to the application from a back-end core script library by implementing additional back-end functionality identified in a back-end project specific script library (i.e., adding custom actions to JSP pages from custom tag libraries developed for specific projects to implement additional custom actions not provided by the built-in tag libraries, see pages 27-46). While Sun does not specifically disclose that the custom tag libraries are categorized as front-end and back-end, it is well known in the art that good software design following the principle of modularity dictates that software systems should be divided into a front-end and a back-end when applicable. Therefore, it is obvious that the custom tag libraries can be classified as providing front-end or back-end functionalities.

As per claim 18, Saimi et al. in view of Sun teaches the computer-readable medium of claim 15, which has been addressed. Saimi et al. further teaches that the template suite comprises at least one front-end template providing functionality directed to presenting and managing data (i.e., the JSP templates used to generate JSP components that generates HTML pages to present and manage data retrieved from the database, see [0074], [0076], and [0085]).

As per claim 19, Saimi et al. in view of Sun teaches the computer-readable medium of claim 15, which has been addressed. Saimi et al. further teaches that the template suite comprises at least one front-end template providing functionality directed to presenting database records (i.e., the JSP templates used to generate JSP components that generates HTML pages to present data retrieved by Beans from a database, see [0074], [0076], and [0085]).

As per claim 20, Saimi et al. in view of Sun teaches the computer-readable medium of claim 15, which has been addressed. Saimi et al. further teaches that the template suite comprises at least one front-end template providing functionality directed to coordinating transactions among different components of an application (i.e., the servlet templates used to generate servlet components that executes requests from the client by issuing requests to the corresponding Bean and passing the Bean to the JSP so that data gathered from the Bean can be retrieved and presented, see [0074], [0076], and [0080]).

Conclusion

16. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

- Hossain et al. (US 5671415) is cited to teach a system and method for facilitating software development.
- Sonderegger, (US 5761499) is cited to teach a method for managing globally distributed software components.
- Fowlow et al. (US 5860004) is cited to teach a code generator for applications in distributed object systems.
- Buxton et al. (US 5970252) is cited to teach a method and apparatus for loading components in a component system.
- Cohen (US 6263352 B1) is cited to teach automated web site creation using template driven generation of active server page applications.

- Huck et al. (US 7111231) is cited to teach a system and methodology for dynamic application environment employing runtime execution templates.
- Wong et al. (US 2002/0080200 A1) is cited to teach a method and apparatus for implementing a web application.
- Gunjal et al. (US 2003/0221184 A1) is cited to teach a template-based application development system.
- Curry et al. (US 2003/0233631 A1) is cited to teach a web services development method.
- Kasravi et al. (US 2004/0172612) is cited to teach a method of software reuse.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Jue S. Wang whose telephone number is (571) 270-1655. The examiner can normally be reached on M-F 9:00 am - 5:00pm (EST).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Xiao Wu can be reached on (571) 272-7761. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

J.W.
2/2/2007

[illegible]
XIAO WU
SUPERVISORY PATENT EXAMINER